Stop-On-Stall

You can enable the Stop-on-Stall function with the **FSD1** command. The move will terminate, without any delay, as soon as a stall is detected This function works either in Motor Step or Encoder Step mode.

CAUTION

Disabling the Stop-on-Stall function with the FSDØ command will allow the AX to finish the move regardless of a stall detection, even if the load is jammed. This can potentially damage user equipment.

The **FSD1** command is valid only if the Enable Stall Detection (**FSH1**) command has been issued.

The Stop-on-Stall function depends on the setting for backlash (set with the **DW** command) to give optimum operation. The factory default setting for backlash is 0 motor steps. **If you mount the encoder on the motor, you should leave this parameter set to zero for the most accurate response.**

Command	Description
DW1ØØ	Sets backlash value to 100 steps.
ER4ØØØ	Sets encoder resolution to 4,000 steps/rev.
FSH1	Enables stall detect.
FSD1	Enables stop on stall.

Stall detection does not occur until the error exceeds the dead band backlash (set with the **DW** command). Consequently, if the indexer does not see an encoder pulse after moving the motor 100 steps, the AX will detect a stall and stop the motor immediately.

Determining Backlash

You can measure the actual backlash with the following procedure. The idea is to move in one direction, stop, and make a series of one-step moves in the opposite direction. No change in encoder position occurs while the indexer takes up the backlash. The number of motor steps counted before any encoder counts are received is the measure of the backlash. *NOTE: When the encoder is mounted directly to the motor, this is mainly a magnetic backlash (or hysteresis), otherwise, gear backlash is also measured.*

Move the motor in one direction and clear the position counters with the following commands.

Command	Description
FSBØ	Sets indexer to motor step mode
MN	Sets Mode normal
A1Ø	Sets acceleration to 10 revs/sec^2
V1	Sets velocity to 1 rps
D64ØØ	Sets distance to 6400 steps
G	Executes the move (Go)
PZ	Sets absolute position counter to zero

Now execute a series of one-step moves and report both motor and encoder position each time:

Command	Description
D1	Sets distance to 1 step
PS	Pauses execution of the following commands until the indexer receives the Continue (**C**) command
L	Causes the sequence to repeat
G	Executes the move (Go)
1LF	Sends a line feed
1PR	Reports absolute motor position in motor steps
T1.Ø	Pauses the motor for 1 second
1LF	Sends a line feed
1PX	Reports the number of encoder pulses (steps) that the encoder has received since the Position Zero (**PZ**) command was issued
T1.Ø	Pauses the motor for 1 second
N	Ends the loop
C	Clears pause and executes the move

While this command string is running, you may compare the position reports from the Position Report (**PR**) command and the Report Absolute Encoder Position (**PX**) command. The **1PR** report represents the number of motor steps traveled, and the **1PX** report represents the number of encoder steps traveled. When the response from the **1PX** command changes from 0 to 1, note the response from the **1PR** command. This **1PR** command report is the actual backlash of your system.

To use the AX's stall detection function, set the Dead Band Window (**DW**) command equal to the value of the backlash determined above and then Enable Stall Detect (**FSH1**).

Output-On-Stall

You can select the Output-on-Stall function with the Turn on Output #1 on Stall Detect (**FSE**) command. This is useful for signaling other components in your system that a stall has occurred.

If you enter the **FSE1** command, the AX programmable Output #1 goes on (current flows) when a stall is detected and remains on until a new move begins. This command is valid only if the Stall Detection has been enabled (**FSH1**), and if the AX is set to Encoder Step Mode (**FSB1**).

Command	Description
MN	Sets the system in the preset mode
FSB1	Enables encoder mode
DW1Ø	Selects backlash to be 10 motor steps
FSH1	Enables stall detect function
FSD1	Stops motor if a stall is detected
FSE1	Turns on Output 1 if stall is detected
A2	Sets acceleration to 2 revs/sec^2
V.1	Sets velocity to 0.1 rps
D128ØØ	Sets distance to 12,800 encoder steps
G	Execute the move (Go)

While the motor is moving, you can cause a stall by holding the shaft. If you can not manually stall the motor, you can simulate a stall by carefully disconnecting the +5V encoder lead from pin #1 on the AX encoder connector. When the stall occurs, Output 1 is turned on and the motor stops (this signals you that the motor has stalled). The motor will come to a stop.

Multi-Axis Stop

On a multi-axis AX system, you may wish to have all axes stop motion if a stall is detected on any axis. You can select this function via the Kill Motion on Trigger 3 (**FSF**) command. When selected with the **FSF1** command, a signal on the Trigger 3 input terminates the move immediately, thus functioning as a remote stop input.

Follow the following steps to set up your AX system to terminate the multi-axis moves when a stall is detected on any axis:

1. Set the first AX unit to Turn on Output 1 on Stall (**FSE1**)
2. Connect Output 1 (pin 15) on the first AX unit to Trigger 3 (pin 11) on all the other AX units in the daisy-chain.

Output on Position Loss

The Turn on Output #2 When Within Dead Band (**FSG**) command allows the AX to signal other system components when the motor is within the dead band. This command is valid only if Stall Detection (**FSH1**) has been enabled; it will have no effect otherwise.

At the end of the move, if the motor is within the specified dead band (**DB**), Output #2 will be turned on.

Command	Description
MN	Sets the system in the preset mode
FSB1	Sets indexer to encoder step mode
FSC1	Enables position maintenance
DB1Ø	Selects dead band at 10 encoder steps (Position Maintenance is activated if the motor's end-of-move position is off by more than 10 encoder steps.)
FSH1	Enables stall detect function
FSG1	Turns on Output 2 if the motor's end-of-move position is within the 10 encoder step dead band range
A2	Sets acceleration to 2 revs/sec^2
V5	Sets velocity to 5 rps
D5ØØ	Sets distance to 500 steps
G	Execute the move (Go)

Program Control

Triggers

You can use the Wait for Trigger (**TR**) command to specify a configuration of trigger conditions to be matched before executing a sequence of buffered commands. The trigger inputs are TRIG 1, TRIG 2, and TRIG 3. The three possible conditions you can specify are as follows:

- 1 = Wait for the trigger input to be high (no current flows)
- Ø = Wait for the trigger input to be low (current flows)
- X = Ignore the trigger input

Command	Description
MN	Sets unit to Preset mode
A2	Sets acceleration to 2 revs/sec^2
V5	Sets velocity to 5 rps
D256ØØ	Sets distance to 25,600 steps
TRØ1X	Waits for Trigger Input 1 to turn on and Trigger Input 2 to turn off (ignores Trigger Input 3)
G	Executes 4,000-step move (Go)

The move will not be executed until current flows on the TRIG 1 input and no current flows on the TRIG 2 input.

The **TS** command is useful for checking the status of the trigger inputs when it appears as though execution is being halted by the **TR** command and all conditions for matching the trigger input configuration defined by the **TR** command appear to be met. It may also be used to initiate external actions by monitoring the trigger inputs manually with a computer controlling the AX.

Delays

You can use the Time Delay (**T**) command to halt the operation of the indexer function for a preset time. If you are in the Continuous Mode (**MC**), you may use the Continous Time (**CTM**) command to run the motor at continuous velocity for a set time, then change to a different velocity.

In the Preset mode (**MN**), the motor finishes the move before the indexer executes the time delay.

Command	Description
PS	Waits for the AX to receive a Continue (**C**) command before executing next command
A7	Sets acceleration to 7 revs/sec^2
V6	Sets velocity to 6 rps
D256ØØ	Sets distance to 25,600 steps
G	Moves motor 25,600 steps
T5.Ø	Waits 5 seconds after the move ends
H	Changes motor direction
G	Moves motor 25,600 steps in the opposite direction
C	Cancels Pause and executes the move

Loops

This section discusses methods of establishing loops in the programs you write for you application. Loops can be created individually or nested within each other.

Single Loops

You may use the Loop (**L**) command to repeat certain programs the one provided below. All the commands between the **L** command and the **N** command are looped (repeated) the number of times indicated in the **L** command. You can use the Immediate Pause (**U**) command to temporarily halt execution of the loop while it is in progress. *NOTE: The **U** command does not work in continuous mode.* To resume loop execution, issue the Continue (**C**) command.

Command	Description
PS	Pauses command execution until the indexer receives a Continue (**C**) command
MPI	Sets mode to incremental
MN	Sets move to normal mode
A5	Sets acceleration to 5 revs/sec^2
V5	Sets velocity to 5 rps
L5	Performs the Loop 5 times
D128ØØ	Sets distance to 12,800 steps
G	Executes the move (Go)
T2.Ø	Delays 2 seconds after the move
N	Ends Loop
C	Initiates command execution

The motor moves a total of 64,000 steps (a succession of five 12,800-step moves with a 2-second pause between each move).

Nested Loops

The example below shows how you can nest a small loop inside a major loop. The only way you can nest a loop is by using the Constant Velocity Loop (**CL**) and End Loop (**CN**) commands in the continuous mode (**MC**). You may nest a constant velocity loop within a standard loop, but not within another CL loop. The following sequence turns on output 1, then output 2, then turns then both off. You can issue the Stop Loop (**Y**) command to terminate both loops.

Command	Description
P	Pauses execution until C command
MC	Sets move to continuous mode
V1	Sets velocity to 1 rps
L10	Loops 10 times
CL5	Constant velocity loop 5 times
CO1X	Turns on output 1
CTM1	Waits 1 second
COX1	Turns on output 2
CV2	Changes velocity to 2 rps
COØØ	Turns off both outputs
CV1	Changes velocity to 1 rps
CN	Ends continuous velocity nested loop
CVØ	Changes velocity to 0 (stop)
G	Executes move
H	Changes direction
N	Ends outer loop
C	Cancels pause and executes move

Nested Loop

Outer Loop

POBs (Programmable Output Bits)

You can turn the programmable output bits (OUT 1 and OUT 2) on and off using the Output (**O**) command. You can use the outputs to signal remote controllers, turn on LEDs, sound buzzers, etc. A one (1) turns on a given output, a zero (0) turns the output off, and an X leaves the output unchanged. The outputs conduct current when they are on and do not conduct when they are off (see the **O** command description in Chapter 5, Software Reference).

Command	Description
MN	Set move to Mode Normal
PS	Pauses execution until indexer receives a Continue (**C**) command
A1Ø	Sets acceleration to 10 revs/sec^2
V5	Sets velocity to 5 revs/sec
D384ØØ	Sets move distance to 38,400 steps
OØ1	Sets programmable output 1 off and output 2 on
G	Executes the move (Go)
OXØ	After the move ends, leaves output 1 unchanged and sets output 2 off
C	Cancels the Pause and executes the move

Move Completion Signal

When you complete a move, you may use the AX's programming capability to signal the end of the current move. In a preset move, you may use one of the following commands:

- **LF** Line feed (see example #1)
- **CR** Carriage return (see example #2)
- **O** Output command (see example #3)
- **"** Quote command (see example #4)

Example #1

Command	Description
PS	Pauses execution until indexer receives a Continue (**C**) command
MN	Sets move to normal mode
A2	Sets acceleration to 2 revs/sec^2
V2	Sets velocity to 2 rps
D384ØØ	Sets distance to 38,400 steps
G	Executes the move (Go)
1LF	Sends a line feed over the RS-232C interface
C	Cancels the Pause and executes the move

The motor moves 38,400 steps. When you complete the move, the unit issues a line feed from the AX to the host over the RS-232C interface.

Example #2

Command	Description
PS	Pauses execution until indexer receives a Continue (**C**) command
MN	Sets move to normal mode
A2	Sets acceleration to 2 revs/sec^2
V2	Sets velocity to 2 rps
D384ØØ	Sets distance to 38,400 steps
G	Executes the move (Go)
1CR	Sends a carriage return
C	Cancels the Pause and executes the move

The motor moves 38,400 steps. When the AX completes the move, it issues a carriage return to the host over the RS-232C interface.

Example #3

Command	Description
PS	Pauses execution until indexer receives a Continue (**C**) command
MN	Sets move to normal mode
A2	Sets acceleration to 2 revs/sec^2
V2	Sets velocity to 2 rps
D384ØØ	Sets distance to 38,400 steps
G	Executes the move (Go)
O1X	Turns on Output 1
C	Cancels the Pause and executes the move

The motor moves 38,400 steps. When the AX completes the move, Output 1 is turned on.

Example #4

Command	Description
PS	Pauses execution until indexer receives a Continue (**C**) command
MN	Sets move to normal mode
A2	Sets acceleration to 2 revs/sec^2
V2	Sets velocity to 2 rps
D384ØØ	Sets distance to 38,400 steps
G	Executes the move (Go)
"DONE	Sends the message, DONE, to the terminal
C	Cancels the Pause and executes the move

The motor moves 38,400 steps. When you complete the move, the AX issues the DONE message to the host over the RS-232C interface.

Sequences

A sequence is a series of commands. These commands are executed in their programmed order whenever the sequence is run. Immediate commands cannot be stored in a sequence, just as they cannot be stored in the command buffer. Only buffered commands may be used in a sequence.

Sequence Programming

The AX has 2,000 bytes of non-volatile memory (EEPROM) to store up to 7 sequences. Each sequence may have up to 256 characters (including delimiters). Note that one sequence cannot borrow any unused portion of another sequence's allocated memory. Use the following commands to define, erase, and run sequences:

Command	Description
XD	Starts sequence definition
XE	Deletes sequence from EEPROM
XP	Sets Power-up Sequence Mode
XQ	Sets/resets interrupted Run mode
XR	Runs a sequence
XRP	Runs a sequence with a pause
XT	Ends sequence definition
XU	Uploads sequence
XZ	Sets power-up sequence to zero

The commands that you enter to define a sequence are presented vertically in the examples below. This was done to help you read and understand the commands. When you are actually typing these commands into your terminal, they will be displayed horizontally.

It is a good practise to erase the sequence with the **XE** command before defining the sequence with the **XD** command. To begin the definition of a sequence, enter the **XD** command immediately followed by sequence identifier number (1 to 7) and a delimiter (pressing the space bar or the carriage return key). The **XT** command ends the sequence definition and automatically loads the sequence into the AX's EEPROM. All commands entered after the **XD** command and before the **XT** command are executed when the sequence is run. An example is provided below.

Command	Description
XE1	Erases Sequence 1
XD1	Begins definition of sequence 1
MN	Sets move to normal mode
A2	Sets acceleration to 2 revs/sec^2
V1Ø	Sets velocity to 10 rps
D128ØØ	Sets distance to 12,800 steps
G	Executes the move (Go)
H	Reverses direction
G	Executes the move (Go)
XT	Ends definition of sequence 1
XR1	Runs Sequence 1

Once you define a sequence, it cannot be redefined until you delete it. You can delete a sequence from EEPROM by entering the **XE** command immediately followed by a sequence identifier (1 to 7) and a delimiter. You may then redefine that sequence into EEPROM . You can issue the Sequence Status Definition (**XSD**) command (preceded by a device address) to verify if the last sequence definition was successful. The possible responses seen on your terminal are *0*, *1*, or *2*. A *0* means the sequence was successfully defined. A *1* means the sequence already exists with the number you have specified. A *2* means there was not enough space in the sequence buffer for that sequence.

To check the status of a sequence, issue the Sequence Status (**XSS**) command. This command must be preceded by a device address and followed immediately by the number (1 to 7) of the sequence and a delimiter. The possible responses are *0* (Empty), *1* (Bad checksum), or *3* (O.K.).

If you wish to check the contents of a sequence, enter the **XU** command. For example, issuing the **1XU1** command causes the AX to send the contents of sequence number 1 to the computer terminal's screen. The 1 preceding the **XU** command is the device address which must be present since the **XU** command is a *device-specific* command. We are assuming in this example that the AX is set up at device address 1.

Sequence Selection

After you define the sequences over the RS-232C interface, you can execute the sequences by using one of the following modes of operation:

- *Stand-alone Operation*
- *Host Computer Operation*
- *Programmable Logic Controller (PLC) Operation*

Stand-alone Operation

This section explains and provides examples of how to store sequences to be automatically executed when you power-up the system, or executed by remote switches. You will first have to define the sequences into the AX non-volatile memory. You will need a computer or PLC with RS-232C communication capabilities for programming the AX .

Power-up Sequence Execution

A single, pre-defined sequence may be executed on power-up by issuing the **XP** command immediately followed by a sequence identifier number (1 to 7) and a delimiter. For example, if an **XP1** command was entered, sequence #1 would be executed the next time the AX was powered up. Sequence #1 would continue to execute on each subsequent power-up until you issue an **XZ** command or a new **XP(0-9)** command. Once the pre-defined sequence is finished executing, the AX returns control to the RS-232C communications port. If no sequence had been defined as Sequence 1, control would automatically go to the RS-232C communications port.

The **XP** command may be issued at any time, except during sequence definition. Once an **XP** command is entered, it is automatically saved in the AX's non-volatile memory.

To determine which, if any, sequence will be executed on power-up, issue the Sequence Status Power-up (**XSP**) command.

Using Remote Sequence Inputs

One method of executing sequences is performed by issuing the **XP9** command. The **XP9** command will not cause any sequence to be immediately executed. Once you issue the **XP9** command, the AX reads the sequence select inputs (Pins 12-14 on AX I/O connector) every time it is powered-up, or every time the **Z** (software reset) command is issued. The status of the sequence select inputs will be interpreted by the AX as a sequence number (see Table 4-2). If, at the time of power up, the number represented by the sequence select inputs is a valid pre-defined sequence (numbers 1 to 7), the AX will automatically execute that sequence. When this first sequence is finished executing, the AX will once again read the sequence select inputs and execute the next valid sequence number present on these inputs. The AX will continue to execute sequences in this fashion until you issue an **S** (stop) or a **K** (kill) command, or if an end-of- travel limit is encountered. If the AX reads the number 8, or any number that has no sequence defined, it will wait until the status of the inputs changes to a valid, pre-defined sequence.

Sequence	SEQ 1	SEQ 2	SEQ 3
1	ON	ON	ON
2	OFF	ON	ON
3	ON	OFF	ON
4	OFF	OFF	ON
5	ON	ON	OFF
6	OFF	ON	OFF
7	ON	OFF	OFF
8*	OFF	OFF	OFF
* Non-valid sequence	OFF = open switch (not pulled to ground) ON = closed switch (pulled to ground)		

Table 4-2. Sequence Select Inputs

When in the XP9 mode, it is possible to cause the AX to pause between sequence execution. This is done with the **XQ1** command. When the **XQ1** command is present within a sequence, the AX will pause after the execution of the sequence and will wait for all sequence select inputs to be OFF (see sequence #8 in Table 4-2). The AX will then read the status of the sequence select inputs and execute the corresponding sequence number. This interrupted run mode will continue until you issue the **XQØ** command.

You can also program the AX to read the sequence select lines on power-up with the **XP8** command. This command will cause the AX to execute the first valid sequence number it reads on the sequence select inputs after power-up (exactly like the **XP9** command). It differs from the **XP9** command in that it returns control to the RS-232C communications port after it finishes executing the first sequence, rather than reading the sequence select lines again and executing another sequence.

Sample Applications and Commands

The following are step-by-step procedures demonstrating how to run sequences. Using a terminal or a computer, key in the following commands:

STEP 1 Issue the **XP9** command.

STEP 2 Define the following sequences:

Command	Description
XE6	Erases Sequence 6
XD6	Defines Sequence 6
MN	Sets move to normal mode
A1	Sets acceleration to 1 rev/sec^2
V20	Set velocity to 20 rps
D128ØØ	Sets distance to 12,800 steps
G	Executes the move (Go)
XT	Ends Sequence 6 definition

Command	Description
XE7	Erases Sequence 7
XD7	Defines Sequence 7
MN	Sets move to normal mode
A1	Sets acceleration to 1 rev/sec^2
V20	Set velocity to 20 rps
D-448ØØ	Sets distance to 44,800 steps (CCW)
G	Executes the move (Go)
XT	Ends Sequence 7 definition

STEP 3 Verify that your programs were stored properly by uploading each entered sequence (**XU** followed by the number of the sequence). If you receive responses that differ from what you programmed, re-enter those sequences.

STEP 4 Make sure all the sequence select inputs are off (not grounded).

STEP 5 Cycle Power or enter the **Z** command. The AX will go through the normal XP9 mode operation. In XP9 mode, the AX scans the sequence select inputs, looking for a valid sequence according to the binary value of the three inputs (see Table 4-2).

STEP 6 Momentarily turn on (ground) the SEQ 2 input. This will execute sequence number 6. *NOTE: After the sequence has finished executing, the inputs are again scanned to execute another sequence.*

STEP 7 Momentarily turn on (ground) the SEQ 1 input. This will execute sequence number 7.

Host Computer Operation

This section assumes you have successfully communicated with the AX. If you have not verified that your RS-232C communications link is functioning properly, return to Chapter 2, Getting Started, and complete this check before attempting to complete this section.

An IBM-compatible software diskette providing terminal emulation is available from Parker Compumotor. To operate the software, you will need BASICA or GW BASIC programming language programs installed in your computer.

Immediate Sequence Execution

You can execute a sequence by entering the **XR** command immediately followed by a sequence identifier number (1 to 7) and a delimiter. The sequence will be executed immediately after the delimiter.

You can issue the Sequence Status Run (**XSR**) command to verify if the last sequence you issued was executed successfully. The possible responses are as follows:

- ***Ø*** Running
- *Non-Zero* Not running:
  - ***1*** In a loop
  - ***2*** Invalid sequence
  - ***3*** Erased
  - ***4*** Bad checksum

Single-Axis Control

The AX system is capable of single and multiple axis applications. The principles developed for a single-axis system apply as well to multi-axis systems.

Single-Axis Interface Program Example

If you already have BASICA or GW BASIC programming languages on your computer, you may use the following sample program designed to open a serial communication port and send and receive AX commands. The program performs the following steps:

- Executes the first move upon user input
- Waits for a line feed from the AX , which indicates the end of the move.
- Upon user input, executes the second move
- Waits for a line feed from the AX, which indicates the end of the move. It then begins the process again.

This application can be looked on as moving a part out, machining the part, then bringing the part back.

```
1 '            AX.BAS PROGRAM
2 '
3 '
   ***********************************************************
4 ' *                                                          *
5 ' * This program controls the RS232 Communication line to execute 2   *
6 ' * different moves using the AX                             *
7 ' *                                                          *
8 ' *                                                          *
   ***********************************************************
15 OPEN "COM1:9600,N,8,1,RS,CS,DS,CD" AS #1     ' Open Communication port
20 V$ = "": Q$ = "": ECHO$ = "": LF$ = "":      ' Initialize variables
90 CLS
100 LOCATE 12,15
105 PRINT " PRESS ANY KEY TO START THE PROGRAM "
107 V$ = INKEY$: IF LEN(V$) = 0 THEN 100        ' Wait for input from user
120 Z$ = "Z"                          ' Reset the AX indexer
122 PRINT #1,Z$;
124 Q$ = INPUT$(2,1)
900 '
     *********************************************************
901 ' *                                                        *
902 ' *  Line 1000-1060 sends a move down to the first AX.  Computer   *
903 ' *  waits for the Line Feed from the AX indicating that the motor *
904 ' *  has finished its move.  Computer will not command second AX to move *
905 ' *                                                        *
906 ' *                                                        *
     *********************************************************
```

```
1000 MOVE1$ = "1A1 1V2 1D25,600 1G 1LF "          ' Define move for Axis 1
1005 CLS
1007 LOCATE 12,15: PRINT " DOING MOVE 1 "
1010 PRINT #1,MOVE1$                          ' Move axis 1.
1015 ECHO$ = INPUT$(23,1)                     ' Read echoes from AX
1020 LF$ = INPUT$(1,1)                        ' Wait for line feed from AX
1040 IF LF$ <> CHR$(10) GOTO 1020                 ' indicating end of move.
1045 CLS
1047 LOCATE 12,15
1050 PRINT "MOVE 1 DONE"                      ' Let user know axis 1 is done
1060 LOCATE 15,15: PRINT " PRESS ANY KEY TO GO ON TO SECOND MOVE "
1070 V$ = INKEY$: IF LEN(V$) = 0 THEN 1060
1900 ' *******************************************************************
1901 ' *                                                                 *
1902 ' * After axis one is done, we request that you press any key to go on *
1903 ' * to the second move. In real application, we would expect you to   *
1904 ' * go ahead with the process and work on the part before going on to *
1905 ' * next move. (i.e. Activate a punch)                                *
1906 ' *                                                                 *
1907 ' * Now that first move is finished, we go on to move #2.            *
1908 ' * AX also prints a line feed after finishing the second move.      *
1909 ' * As soon as computer receives the line feed from AX, program will *
1910 ' * go back to the first move.                                       *
1911 ' *                                                                 *
1912 ' *******************************************************************
2000 MOVE2$ = "A10 V5 D-64000 G H G 1LF"
2005 CLS
2007 LOCATE 12,15: PRINT " DOING MOVE 2 "
2010 PRINT #1,MOVE2$
2015 ECHO$ = INPUT$(25,1)
2020 LF$ = INPUT$(1,1)
2040 IF LF$ <> CHR$(10) GOTO 2020
2045 CLS
2047 LOCATE 12,15
2050 PRINT "MOVE 2 DONE "
2060 FOR I = 1 TO 1000: NEXT I
2070 GOTO 20                              ' Go back to beginning of program.
```

Multi-Axis Control

Device-specific commands require that a device address precede them. The AX will not execute a device-specific command if there is no device address preceding the command, or if the device address setting in the AX does not match the address preceding the command. *Universal* commands do not require device identifiers preceding them. A universal command with no device address will be executed regardless of the address setting of the AX. If a device address does precede a universal command, it will only be executed by an AX set to that particular address.

The **E** (Enable RS-232 communication) and **F** (Disable RS-232 communication) commands are useful in a daisy chain for locking out particular indexers from responding to universal commands with no preceding device address.

*NOTE: The **F** command will keep the AX from executing any commands (except the **E** command) sent to it over the RS-232C interface, but will not prevent the command from being echoed.*

For Example, to lock-out the AX unit set to device address 1, so that universal commands are only executed by the AX's set to address 2 and 3, the following step is performed:

- Send the **1F** command over the RS-232C communication line, locking out the AX at device address 1.

All universal commands (with no preceding device address) will now be executed only by the AX's at Device addresses 2 and 3. Entering a **2F** command in addition to **1F** command would allow only the AX at device address 3 to execute universal commands with no preceding device address. This eliminates the need to precede every command intended for a specific AX (in this case, the AX at device address 3) with a device address. Sending an **E** command over the RS-232C line will re-enable all drives previously disabled with the **F** command. Preceding the **E** command with a device address will re-enable only the AX set to that particular device address.

When using the **XU** command to upload the contents of a specified sequence from an AX in a daisy-chain, it is necessary to disable all the drives in the chain that come after the drive being queried with the **F** command (as described above). This will prevent subsequent drives from executing the commands being sent back to the terminal. See the **XU** command description in Chapter 5, Software Reference .

For daisy-chain wiring instructions, refer to Chapter 2, Getting Started.

Sample Application and Commands

Example: Three indexers are on an RS-232C daisy chain. Send the following commands:

Command	Description
MN	Sets unit to Preset mode
A5	Sets acceleration to 5 revs/sec^2 for all three indexers
V1Ø	Sets velocity to 10 rps for all three indexers
1D64ØØ	Sets Axis 1 distance to 6,400 steps
2D128ØØ	Sets Axis 2 distance to 12,800 steps
3D192ØØ	Sets Axis 3 distance to 19,200 steps
G	Moves all axes.

Unit 1 moves 6,400 steps, unit 2 moves 12,800 steps, and unit 3 moves 19,200 steps. All three units use the same acceleration and velocity rates. Units 1, 2, and 3 will start at about the same time (2mS apart, due to a propagation delay of 2 characters over the RS-232C interface).

Multi-Axis Interface Program Example

The following program is very similar to AX.BAS, except this program controls 2 AX's on a daisy chain. This program assumes the device address of 2 AX's to be 1 and 2 respectively. The program does the following:

- Executes the first move upon user input
- Waits for a line feed from the LX drive, which indicates the end of the move.
- Upon user input, executes the second move on axis 2
- Waits for a line feed from the second AX, which indicates the end of the move. It then begins the process again.

```
1 '            AX2.BAS PROGRAM
2 '
3 '
   ***************************************************************
4 ' *                                                             *
5 ' * This program controls the RS232 Communication line to execute 2  *
6 ' * different moves using 2 AX units..                          *
7 ' *                                                             *
8 ' *                                                             *
   ***************************************************************
15 OPEN "COM1:9600,N,8,1,RS,CS,DS,CD" AS #1      ' Open Communication port
20 V$ = "": Q$ = "": ECHO$ = "": LF$ = "":       ' Initialize variables
90 CLS
100 LOCATE 12,15
105 PRINT " PRESS ANY KEY TO START THE PROGRAM "
107 V$ = INKEY$: IF LEN(V$) = 0 THEN 100         ' Wait for input from user
120 Z$ = "Z "                          ' Reset the AX indexer
122 PRINT #1,Z$;
124 Q$ = INPUT$(2,1)
```

```
900 '
    ************************************************************
901 ' *                                                          *
902 ' *  Line 1000-1060 sends a move down to the first AX.  Computer  *
903 ' *  waits for the Line Feed from the AX indicating that the motor *
904 ' *  has finished its move.                                  *
905 ' *                                                          *
906 ' *                                                          *
    ************************************************************
1000 MOVE1$ = "1A1 1V2 1D25,600 1G 1LF "         ' Define move for Axis 1
1005 CLS
1007 LOCATE 12,15: PRINT " MOVING AXIS 1 "
1010 PRINT #1,MOVE1$                             ' Move axis 1.
1015 ECHO$ = INPUT$(22,1)                        ' Read echoes from AX
1020 LF$ = INPUT$(1,1)                           ' Wait for line feed from AX
1040 IF LF$ <> CHR$(10) GOTO 1020                ' indicating end of move.
1045 CLS
1047 LOCATE 12,15
1050 PRINT "AXIS 1 FINISHED ITS MOVE "           ' Let user know axis 1 done
1060 LOCATE 15,15: PRINT " PRESS ANY KEY TO MOVE SECOND AXIS "
1070 V$ = INKEY$: IF LEN(V$) = 0 THEN 1060
1900 ' ************************************************************
1901 ' *                                                          *
1902 ' * After axis one is done, we request that you hit any key to go on  *
1903 ' * to second move.  In real application, we would expect you to  *
1904 ' * go ahead with the process and work on the part before going on to  *
1905 ' * next move.  (i.e.  Activate a punch)                     *
1906 ' *                                                          *
1907 ' * Now that first axis finished its move, we go on to move axis 2.  *
1908 ' * Second AX also prints a line feed after finishing the move.  *
1909 ' * As soon as computer receives the line feed from AX, program will  *
1910 ' * go back to the first move.                               *
1911 ' *                                                          *
1912 ' ************************************************************
2000 MOVE2$ = "2A1 2V2 2D6,400 2G 2LF "
2005 CLS
2007 LOCATE 12,15: PRINT " AXIS 2 MOVING "
2010 PRINT #1,MOVE2$
2015 ECHO$ = INPUT$(22,1)
2020 LF$ = INPUT$(1,1)
2040 IF LF$ <> CHR$(10) GOTO 2020
2045 CLS
2047 LOCATE 12,15
2050 PRINT "AXIS 2 FINISHED ITS MOVE "
2060 FOR I = 1 TO 1000: NEXT I
2070 GOTO 20                     ' Go back to beginning of program.
```

PLC Operation

You can use a PLC to execute 7 different sequences that are stored in the AX non-volatile memory. Three outputs from the PLC can be used to execute sequences, and two inputs to the PLC can be used to monitor AX outputs.

PLC Connections

Assuming your PLC accepts open-emitter outputs, connect the inputs and outputs as shown in Figure 4-4. If not, contact a Compumotor Applications Engineer at (800) 358-9070.

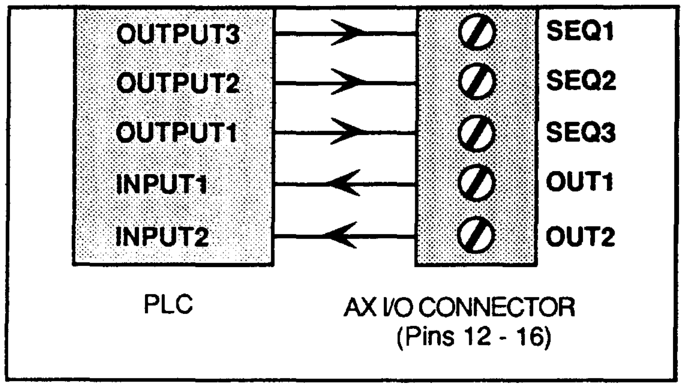

Figure 4-4. PLC Connections

Scanning for Sequence Execution

Changing the BCD values of sequence input lines results in a new sequence being run that corresponds to the new value. As indicated in Table 4-2, the configuration of the values issued determines which sequence the indexer will run. For example, turning on SEQ2 and SEQ3 and turning off SEQ1 executes Sequence 2.

Issuing the **XP8** or the **XP9** commands causes the AX to scan the sequence select inputs and execute the fist valid sequence it encounters (on power-up). When in XP8 mode, the AX returns control to the RS-232C port after executing the first valid sequence. When in the XP9 mode, the AX will continue scanning inputs and executing valid sequences until you issue an **S** or a **K** command.

When you issue the **XP** command identifying sequences 1 - 7, this over-rides the sequence input configuration used when you issue the **XP8** and the **XP9** commands.

The Scan (**SN**) command determines how long the sequence select input must be maintained before the indexer executes the program. This is a debounce time.

Sample Applications and Commands

This section provides step-by-step procedures to run sequences from your PLC. First, you need to enter the programs into the AX drive. You will need a terminal or a computer with RS-232C communication capability. You need to define the sequences before you can execute them with your PLC's BCD outputs.

Using a terminal or a computer, key in the following commands:

STEP 1 Issue the **XP9** command.

STEP 2 Define the following sequences:

Command	Description
XE1	Erases Sequence 1
XD1	Defines Sequence 1
MN	Sets move to normal mode
A1	Sets acceleration to 1 rev/sec^2
V10	Set velocity to 10 rps
D64ØØ	Sets distance to 6,400 steps
G	Executes the move (Go)
XT	Ends Sequence 1 definition

Command	Description
XE2	Erases Sequence 2
XD2	Defines Sequence 2
MN	Sets move to normal mode
A1	Sets acceleration to 1 rev/sec^2
V10	Set velocity to 10 rps
D32ØØ	Sets distance to 3,200 steps
G	Executes the move (Go)
XT	Ends Sequence 2 definition

Command	Description
XE3	Erases Sequence 3
XD3	Defines Sequence 3
MN	Sets move to normal mode
A1	Sets acceleration to 1 rev/sec^2
V10	Set velocity to 10 rps
D128ØØ	Sets distance to 12,800 steps
G	Executes the move (Go)
XT	Ends Sequence 3 definition

Command	Description
XE4	Erases Sequence 4
MN	Sets move to normal mode
XD4	Defines Sequence 4
A1	Sets acceleration to 1 rev/sec^2
V10	Set velocity to 10 rps
D-448ØØ	Sets distance to 44,800 steps (CCW)
G	Executes the move (Go)
XT	Ends Sequence 4 definition

STEP 3 Verify that your programs were stored properly by uploading each entered sequence (**XU** command preceded by the device address and followed by the number of the sequence). If you receive responses that differ from what you programmed, re-enter those sequences.

STEP 4

Cycle Power or enter the **Z** command. The AX will go through the normal XP9 mode operation. In XP9 mode, the AX scans the sequence select inputs and selects and executes the first available valid sequence according to the binary value of the three inputs (see Table 4-2). After the sequence is executed, the inputs are again scanned to execute another sequence. The AX will continue to execute sequence in this fashion until you issue an **S** or a **K** command, or if an end-of-travel limit is encountered.

STEP 4

To run sequence #1, ground sequence inputs 1, 2, and 3 at the same time (refer to table 4-2 for sequence input configurations to run other sequences).

Model 72 (Thumbwheel) Operation

The Model 72 and Model 72-I/O units interface with the AX Drive via an RS-232C interface. The Model 72 provides adjustable thumbwheel switches to enter and change AX motion control parameters. These switches allow you to set and modify velocity, acceleration, distance, direction, dwell time, loop count, and other parameters.

If you wish to purchase a Model 72 or Model 72-I/O, contact your Automated Technology Center or Compumotor Distributor. The *Model 72 User Guide* contains all pertinent information for use with the AX.

Tuning Procedure

NOTE: For most applications, tuning an AX Drive to a motor is not necessary.

Should certain aspects of the systems performance come under scrutiny in the area of velocity ripple, resonant frequencies or step-to-step accuracy, the following procedure could offer improved performance.

Velocity ripple is a term used to express a variance of angular speed. It is usually expressed as a percentage of the commanded velocity. This is due to the mechanical inaccuracies of the motor resulting in unequal step sizes. It can also be enhanced by running the motor at its' natural frequency causing resonance. Adding solid inertia to the system can shift or modify the phenomenon but does little to eliminate it. If your system can handle more inertia, consider using viscous dampers.

A 10:1 load to rotor inertia ratio is the maximum value recommended.

To a certain extent, adding friction to the system can actually dampen the effects of both resonance and velocity ripple.

The recommended system frictional load for a step motor is approximately 5% of the motor's maximum torque capability. For motor specifications and speed/torque curves, refer to Chapter 6, Hardware Reference.

The step-to-step accuracy of the step motor should not be confused with the motors absolute accuracy. Absolute accuracy is based on the quality of the mechanical construction of the motor. This means that tuning of an open-loop stepper motor and drive will not improve this spec. On the other hand, the drive electronics are responsible for adjusting current in the two phases of the motor which create the microsteps found between each 1.8° full motor step. The amount of distance each step can vary is referred to as the step-to-step accuracy of the system and it may be possible to improve this by tuning.

The motor should be tuned to its drive while under normal loaded conditions. Tuning prior to this is useless since changes in the load or the system's transmission will result in a new natural frequency for the motor.

Tuning the drive affects the profile of current being sent to the motor. You will need to re-tune the drive if the current setting switches are changed. Refer to Chapter 6 for valid settings for DIP switches 1 through 5.

Controlling motor velocity

When using the AX Drive, the built-in indexer will solve this problem. Using an RS-232C communications device, place the AX in Mode Continuous (**MC**) and use the Acceleration (**A**) and the Velocity (**V**) commands to get the motor moving, then vary the velocity using the Immediate Velocity (**VC**) command.

Command	Description
MC	Set to Mode continuous
A10	Set acceleration to 10 rev/sec
V1	Set velocity to 1 rev/sec
G	Execute the move (Go)
•	
•	
•	
VC2	Set immediate velocity to 2 rev/sec
•	
•	
•	
VC.5	Set immediate velocity to .5 rev/sec

Gauging Motor Resonance

Selecting a method of gauging motor resonance can be accomplished in a number of ways;

TACHOMETER METHOD

Using an oscilloscope to look at the output of a tachometer attached to the motor shaft. The tachometer will output a DC voltage, proportional to speed. This voltage will oscillate around an average voltage when the motor is resonating. The amplitude of this oscillation will be at a maximum when the natural frequency of the motor is located. Using this method your goal is to tune for the lowest oscillation amplitude.

SOUNDING BOARD METHOD

When practicing your tuning skills, an unloaded motor can be placed on a sounding board or a table. When a velocity is commanded which is near the motors natural frequency, the phenomenon will cause vibration which will become audible. Using this method your goals is to tune for the lease amount of vibration.

STETHOSCOPE METHOD

When tuning under loaded conditions the audible vibration caused by the motors natural frequency can be heard by placing the tip of a screw drive against the motor casing and holding the handle of the screw driver close to your ear (as you would a stethoscope). This will also allow you to hear the different magnitudes of vibration caused by the motors natural frequency. Using this method your goal is to tune for the least amount of vibration.

TOUCH METHOD

After some experience with tuning, it is possible to locate a motor's natural frequency by simply placing your fingertips on the motor shaft and adjusting the motor velocity. Once the critical velocity is located, tuning for maximum smoothness can be done in the same way.

CAUTION

Exercise extreme caution whenever you are near any moving part of a mechanical system. Also, be wary of touching the motor, as it is very hot during normal operating conditions.

To tune the drive you will need a small non-conductive screw driver.

Locate the natural frequency of the System

Before you can tune the drive to its motor, you will need to locate the natural frequency of the motor and its load. This can be accomplished using the following procedure:

STEP 1 Let the motor and driver warm up for 30 minutes to an hour.

STEP 2 Locate the tuning pots on the drive you are using. Figure 4-5 below shows the location of the AX Drive tuning pots. Note that the small metal plate held in place with two screws must be removed to reveal the tuning pots.

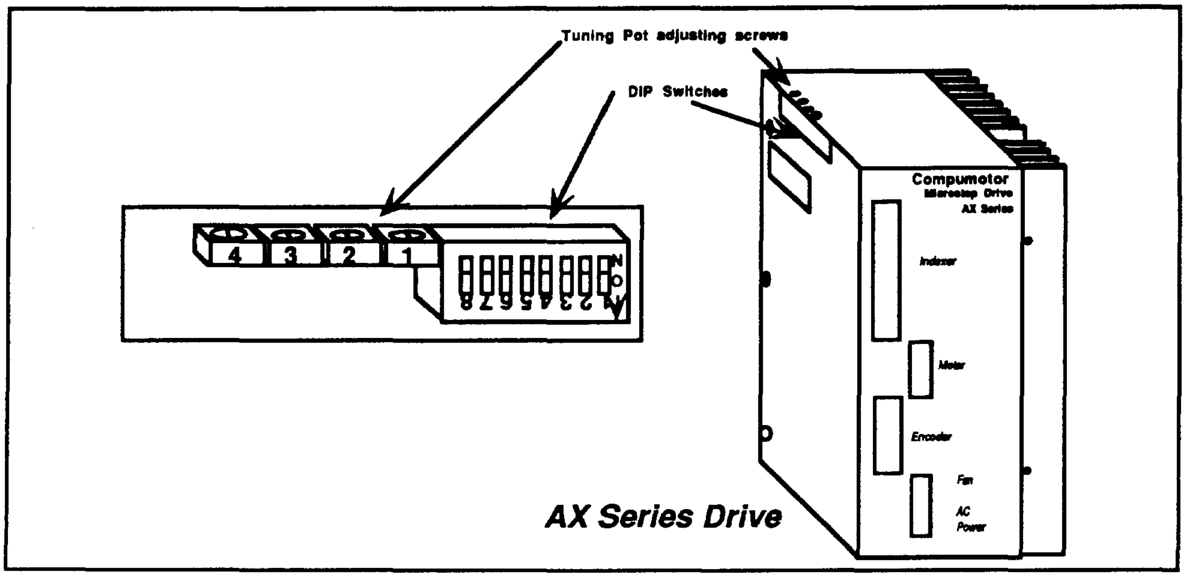

Figure 4-5. AX DIP Switch and Tuning Pot Locations

STEP 3 Position all four tuning pots to the center of their adjustment range.

STEP 4 Start the motor moving at one of the velocities shown below, depending on the motor frame size. These velocities should get you close to the natural frequency of the motor.

57 Series (NEMA 23)	3.6 rps
83 Series (NEMA 32)	2.8 rps
106 Series (NEMA 42)	2.0 rps

STEP 5 Using one of the gauging methods mentioned above, increase and decrease the velocity in small increments (.1 rps and lower) until you have located the velocity where the natural frequency seems to be most prominent.

Adjusting the Drive to the Motor

Now that you have located the natural frequency of your system, the following procedure will adjust the current waveform of the drive to the mechanical characteristics of the system.

STEP 1 Adjust the **current trim** if necessary (pot #1)

The current trim pot allows you to select a current setting that lies between any two dip switch settings. In most applications, adjustment of this setting is not necessary.

STEP 2 Adjust **phase balance** (pot #4).

Adjust this pot for maximum smoothness. This pot adjusts the amplitude of current in phase B to within ±10% of the current in Phase A.

STEP 3 Adjust **phase B offset** (pot #2) and **phase A offset** (pot #3).

Alternately adjust these two pots for maximum smoothness as well. These pots adjust the zero (0) crossing point of the current in each phase.

STEP 4 Re-locate the natural frequency again.

Increase and decrease the velocity in small increments (.1 rps and lower) until you have re-located the velocity where the natural frequency seems to be most prominent.

STEP 5 Repeat steps 1 through 4 until you have maximized the motor's performance.

Selecting a new Wave Shape

All the above adjustments have affected only the total amount of current being sent to the motor and the amount in each phase of the motor with respect to each other. This can be very effective in most applications, but it may be possible to take your tuning procedure one step further by changing the shape of the *current waveform* itself. See Table 4-3.

Waveform	% of 3rd Harmonic
#1	-8%
#2	-6%
#3	-4%
#4	-2%
#5	0%
#6	+2%
#7	+4%
#8	+6%
#9	+8%

Table 4-3. AX Waveform Settings

WAVE SHAPING FOR THE AX DRIVE

STEP 1 Stop motion to the motor.

STEP 2 Use the **WV** (Select Micro-stepping Waveform) command to select a new waveform. The new waveform will take effect immediately. There are 9 different waveforms available and waveform #5 is the default setting.

STEP 3 Re-locate the natural frequency again.

STEP 4 Repeat the section titled **Adjusting the Driver to the Motor**

STEP 5 Repeat steps 1 through 3 in this section until maximum performance is achieved.

STEP 6 Once you have chosen the new waveform, you will need to command the new value every time you power up or the command can be made part of a non-volatile memory sequence that is executed on power up.